L11	L1 and (subscriber near information)	918	L11
L10	L9 and (subscriber near information)	0	L10
L9	l6 and (event and (chang$ or updat$))	24	L9
L8	L6 and (subscriber$ near director$)	0	L8
L7	L6 and subcriber$	0	L7
L6	L5 and l1	25	L6
L5	email near schedul$	64	L5
L4	L3 and (event$ and chang$)	7	L4
L3	L2 and (updat$ near synchroniz$)	7	L3
L2	L1 and (subscriber$ near message$)	966	L2
L1	messag$ near system	18088	L1

END OF SEARCH HISTORY

First Hit

End of Result Set

Generate Collection Print

L29: Entry 2 of 2 File: PGPB Jun 6, 2002

DOCUMENT-IDENTIFIER: US 20020069060 A1
TITLE: Method and system for automatically managing a voice-based communications systems

Detail Description Paragraph:
[0049] A telephone network can connect the retrieval system 305 with the voicemail system 304.

Detail Description Paragraph:
[0064] The system 405 may contain (or be connected to another system that contains) a database 406 of subscriber information such as user ID, passwords and external voice mail service information. The external voice mail service information contains, for example a telephone number which is used to call in to the external voice mail system and the user ID (mailbox number) and password of the person's account (voice mailbox) on external voice mail system. In other variation, this information could be entered by the person 401 at the time he/she makes the telephone call 403.

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: ((synchroniz$ or update) near (voice near message$))

Display: 50 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Friday, February 13, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L36	((synchroniz$ or update) near (voice near message$))	14	L36
L35	(synchroniz$ or update) near (voice near (message$))	14	L35
L34	L32 and (shar$ near directory)	2	L34
L33	L32 and (subscriber near voice near messag$)	3	L33
L32	L31 and updat$	117	L32
L31	L30 and (voice near messag$)	131	L31
L30	l28 and (synchroniz$ or updat$)	240	L30
L29	L28 and (external near mailbox)	2	L29
L28	(voicemail near system)	430	L28
L27	(707/$.ccls.) and L26	11	L27
L26	L25 and level	42	L26
L25	L24 and recover$	42	L25
L24	L22 and (synchronous)	42	L24
L23	L22 and (synchronous near replication)	2	L23
L22	L21 and (secondary or remote)	51	L22
L21	l18 and (data near replication)	53	L21

L20	L18 and (primary near data)	17	L20
L19	L18 and (primary near data) and (secondary near data)	9	L19
L18	l7 and (distribut$ near system)	53	L18
L17	L16 and (storage near device)	20	L17
L16	L14 and secondary	49	L16
L15	L14 ans secondary	1637206	L15
L14	L13 and primary	49	L14
L13	L12 and (low near level)	49	L13
L12	L11 and java	54	L12
L11	l8 and presentation	55	L11
L10	L8 and (presentation near program)	10	L10
L9	L8 and ((low near level) near routine$)	9	L9
L8	L7 and synchroniz$	58	L8
L7	l1 and (three near tier$)	72	L7
L6	L5 and synchroniz$	53	L6
L5	l1 and (three near tiers)	64	L5
L4	L2 and (distribut$ near system)	10	L4
L3	L2 and (distrribut$ near computer)	0	L3
L2	(remote near data near replicat$)	38	L2
L1	(data near replication)	1256	L1

END OF SEARCH HISTORY

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Saturday, February 14, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L36	((synchroniz$ or update) near (voice near message$))	14
☐	L35	(synchroniz$ or update) near (voice near (message$))	14
☐	L34	L32 and (shar$ near directory)	2
☐	L33	L32 and (subscriber near voice near messag$)	3
☐	L32	L31 and updat$	117
☐	L31	L30 and (voice near messag$)	131
☐	L30	l28 and (synchroniz$ or updat$)	240
☐	L29	L28 and (external near mailbox)	2
☐	L28	(voicemail near system)	430
☐	L27	(707/$.ccls.) and L26	11
☐	L26	L25 and level	42
☐	L25	L24 and recover$	42
☐	L24	L22 and (synchronous)	42
☐	L23	L22 and (synchronous near replication)	2
☐	L22	L21 and (secondary or remote)	51
☐	L21	l18 and (data near replication)	53
☐	L20	L18 and (primary near data)	17
☐	L19	L18 and (primary near data) and (secondary near data)	9
☐	L18	l7 and (distribut$ near system)	53
☐	L17	L16 and (storage near device)	20
☐	L16	L14 and secondary	49
☐	L15	L14 ans secondary	1637206
☐	L14	L13 and primary	49
☐	L13	L12 and (low near level)	49
☐	L12	L11 and java	54
☐	L11	l8 and presentation	55
☐	L10	L8 and (presentation near program)	10
☐	L9	L8 and ((low near level) near routine$)	9
☐	L8	L7 and synchroniz$	58
☐	L7	l1 and (three near tier$)	72
☐	L6	L5 and synchroniz$	53

☐	L5	l1 and (three near tiers)	64
☐	L4	L2 and (distribut$ near system)	10
☐	L3	L2 and (distrribut$ near computer)	0
☐	L2	(remote near data near replicat$)	38
☐	L1	(data near replication)	1256

END OF SEARCH HISTORY

L15: Entry 17 of 21 File: USPT Jun 15, 1999

DOCUMENT-IDENTIFIER: US 5913032 A
TITLE: System and methods for automatically distributing a particular shared data object through electronic mail

Abstract Text (1):
A computer system having a facility for concurrently sharing objects or resources is described. The system includes a publish-and-subscribe facility or "Object Exchange," for facilitating sharing among workgroups. When a data object is "published" by a user ("publisher"), the object is sent from that user's computer to other computer users specified by the publisher. Those interested in the published data object (published pages) may elect to receive or "subscribe" to that data. From that point on, the publisher can choose to update the data, such as whenever the published version changes. The "subscribers" of the published pages automatically get updates. Subscribers of a spreadsheet notebook, for instance, would automatically receive pages as they are published. The Object Manager effects actions by posting messages or "forms" to either the local Object Exchange (assuming one is the publisher) or the Object Exchange of others (subscribers). Connectivity for the system is provided by the various Object Exchange engines negotiating forms. Each Object Exchange posts and retrieves forms at polling intervals (which may be set to continuous polling). By exploiting the connectivity of existing transport media (e.g., LANs), the present invention delivers workgroup computing benefits to users, but without imposing a rigid structure which restricts when and how they work.

Brief Summary Text (5):
The first personal computers (PCs) debuted in a corporate world which was dominated by mainframes and mini-computers--computers which handled large, centralized databases for the most part and ran little or no personal productivity software. As a result, these early PCs went largely unused. With the advent of electronic spreadsheets and other personal productivity software (e.g., wordprocessors), however, individual workers were soon empowered to handle their own data processing needs, thus the PC revolution was born.

Brief Summary Text (6):
Early on in the use of PCs, much emphasis was given to autonomy--removing users from the centralized control which was the hallmark of mainframes and mini-computers. Although this increased freedom led to increased productivity gains, users soon realized that they needed services which had been provided by the large centralized systems: shared printers, electronic mail, shared databases, and easy data transfer.

Brief Summary Text (7):
A solution to this problem was forthcoming in the form of Local Area Networks (LANs). A LAN typically comprises several computers connected together as a group. At least one of the computers functions as a "server," providing network services to "clients" (other computers) connected to the network. In this manner, valuable information and resources, including programs, information tables, memory, disk space, printers, and the like, may be shared by several users.

Brief Summary Text (8):

Early LAN systems were patchwork, unreliable systems that offered little more than printer and file sharing. These systems provided no built-in messaging facilities, no database or communications services, or the like. Moreover, PC operating system software (e.g., MS-DOS) took little or no advantage of LANs. Thus throughout most of the past decade, LANs essentially functioned as large storage and printer sharing devices; their true power had yet to be exploited.

Brief Summary Text (10):
The most popular approach employed in this new corporate data processing environment is the "client/server" model. Using a client/server database system, for instance, an end user at a personal computer (client) submits a query to the database server, either directly by using the database or indirectly by using an application. The query is sent over the network to the database server, which processes the query and returns the result. Thus, the client/server model has been recognized as a fast way for deploying database applications in the corporate world. Although client/server database applications have finally begun to exploit LAN messaging services to deliver enhanced productivity, these benefits were not quickly extended to other personal productivity software--namely, spreadsheets and wordprocessors.

Brief Summary Text (11):
As Local Area Networks proliferated, it was recognized that the connectivity that LANs provided to client/server databases could be extended to other applications. The basic approach is that of a "workgroup." The premise of workgroup computing is that by imposing a structure--usually a logical organization consisting of rules-based automatic data distribution--on network-connected PCs, a group of people working together can achieve more than if each individual set out on his or her own and used the network only when necessary.

Brief Summary Text (12):
Currently-available workgroup products are server-based, network-specific implementations. These workgroup products impose a rigid structure on those who use them and do not yield results unless entire organizations commit to them completely. Every user in a workgroup must commit to using the workgroup software in a particular fashion before productivity gains for the group may be realized.

Brief Summary Text (15):
The system of the present invention comprises a computer system providing an Object Exchange module for sharing data objects among application processes (both remote and local). From a user's perspective, the Object Exchange is a publish-and-subscribe facility, which facilitates sharing among workgroups. When a data object, such as one or more spreadsheet pages from a spreadsheet notebook, is "published," it is sent from a desktop computer to other computer users specified by the publisher. Those interested in the published data object (published pages) may elect to receive or "subscribe" to that data. From that point on, the publisher can choose to update the data, such as whenever the published version changes. The "subscribers" of the published pages automatically get updates. Subscribers of a spreadsheet notebook, for instance, would automatically receive pages as they are published.

Brief Summary Text (17):
When a user publishes, issues versions, or mails data objects, the system submits the published or mailed objects to the Object Exchange, for holding them in its outgoing queues. When the Object Exchange polls an account, it transmits all the objects--notebooks, sets of pages, or individual pages--that the user has sent to that account, and picks up all the objects that the user has received to that account since the Object Exchange last polled it. The Object Exchange holds incoming objects in the user's Object Store, ready for use. Thus the Object Store holds copies of shared data objects and, thus, behaves like a database of the shared objects sent to the user. The Object Store makes all of the user's shared

objects continuously available for use in any notebook, whether or not the user is currently connected to messaging services. In a preferred embodiment, shared objects remain in the Object Exchange until deleted.

Brief Summary Text (18):
The Object Exchange is preferably implemented not as an application in itself, but as a middle link between applications and the outside world. In other words, it interacts with the operating system and messaging services to provide workgroup and communication services to applications. The Object Exchange implements an object-sharing API (Application Programming Interface) so that existing applications (clients) can issue commands to the Object Exchange to add workgroup capabilities. Communication between the Object Exchange and a client application is effected through an interprocess communication link, such as available with Windows Dynamic Data Exchange (DDE) or Object-Linking and Embedding (OLE).

Drawing Description Text (4):
FIG. 1C is a block diagram of a multi-user computing environment, such as a client/server system connected via a Local Area Network (LAN), in which the present invention is most preferably embodied.

Detailed Description Text (9):
automatic updating: The process by which published data automatically appears in a subscriber application.

Detailed Description Text (12):
DDE client: An application that initiates a DDE conversation with a named DDE server and makes requests of it.

Detailed Description Text (13):
DDE item: A string that acts like a server-owned variable within a particular DDE topic. In OBEX, every item is classified according to how it can be used, as follows:

Detailed Description Text (17):
DDE server: An application that offers services to other applications via DDE. When a server opens, it registers its server name with the DDE system, making it available to DDE client applications. When the OBEX Enabling Kit is used, OBEX is a DDE server.

Detailed Description Text (18):
DDE topic: A string defining the type of service requested (sent from the DDE client to the DDE server).

Detailed Description Text (19):
hot link: A DDE link where the DDE server sends data to the DDE client whenever the data changes.

Detailed Description Text (20):
LAN mailbox: A directory that is created on a network file server that enables the LAN to store and forward objects and messages.

Detailed Description Text (21):
mail system: Like MCI, MHS, and Windows for Workgroups, mail systems provide messaging and communication services over LANs, WANS, or telephone lines.

Detailed Description Text (22):
manual updating: The process by which published data appears in the subscriber application only when the subscriber requests an update.

Detailed Description Text (27):

object: Any data component packaged and distributed as a message. For example, spreadsheet objects can be notebooks, pages, and sets of pages. Database objects can be results of queries and tables.

Detailed Description Text (32):
publishing and subscribing: When information is sent to other workgroup members, that information is published. Those who receive the information are subscribers.

Detailed Description Text (34):
request: A message from a DDE client to a DDE server to send data, receive data, or execute a command.

Detailed Description Text (35):
server name: A string used by the DDE system to identify the server in a DDE dialog. For many applications, the server name is the same as the file name. For example, the server name for the application OBEX.EXE is "obex".

Detailed Description Text (39):
warm link: A DDE link where the DDE server notifies the DDE client whenever data changes.

Detailed Description Text (47):
While the present invention is operative within a single (standalone) computer (e.g., system 100 of FIG. 1A), the present invention may be embodied in a multi-user computer system, such as the client/server system 130 of FIG. 1C. Specifically, system 130 includes a first computer or server 131 and one or more second computers or clients 150. In an exemplary embodiment, the clients or workstations 150 are connected to server 131 through a computer network 141, which may be a conventional Local Area Network (LAN). Network 141 includes cabling 145 for connecting the server and each workstation to the network. The workstations themselves will be similar to or the same as system 100; additionally, each typically includes a network connector or adapter 143 for receiving the network cable 145, as is known in the art. Server 131 may also be similar to or the same as system 100. Because the server manages multiple resources for the clients, it should preferably include a relatively faster processor, larger mass storage, and more system memory than is found on each workstation.

Detailed Description Text (48):
Overall operation of the system 130 is directed by a networking operating system 137, which may be stored in the server's system memory; in a preferred embodiment, OS 137 includes NetWare.RTM., available from Novell of Provo, Utah. In response to requests from the clients 150, the server 131 provides various network resources and services. For instance, multiple users (e.g., workstations A, B, and C) may view a database table stored in file server storage 139, while another user (e.g., workstation E) sends a document to a network printer (not shown).

Detailed Description Text (52):
The following description will focus on the presently preferred embodiments of the present invention, which are embodied in spreadsheet application software operative in the Microsoft.RTM. Windows environment. The present invention, however, is not limited to any particular application or any particular environment. Instead, those skilled in the art will find that the teachings of the present invention may be advantageously applied to a variety of other applications, including database management systems, wordprocessors, and the like. Moreover, the present invention may be embodied on a variety of different platforms, including Macintosh, UNIX, NextStep, and the like. The present invention is particularly advantageous when applied in those instances where data objects are to be shared. Therefore, the description of the exemplary embodiments which follows is for purposes of illustration and not limitation.

Detailed Description Text (68):
The system of the present invention employs an object exchange architecture, which may be viewed from two perspectives. From the user's perspective, the Object Exchange is a publish-and-subscribe facility, as shown in FIG. 3A, which facilitates sharing among workgroups. When a data object, such as one or more spreadsheet pages from notebook 310, is "published," it is sent from a desktop computer to other computer users specified by the publisher. Those interested in the published data object (published pages 320) may elect to receive or "subscribe" to that data. From that point on, the publisher can choose to update the data, such as whenever the published version changes. The "subscribers" of the published pages 320 automatically get updates. For this example, the spreadsheet notebook of each subscriber (e.g., notebook 330) would automatically receive pages as they are published.

Detailed Description Text (69):
The publish/subscription model is well suited for solving real-world data processing needs. Consider, for instance, a hypothetical company having four sales districts, each having a manager responsible for reporting sales activities. These four managers report to a country manager, who in turn presents a summary report every week to the vice president of Sales for the company. Restated in terms of a publish/subscribe model, the job of all four district managers is to publish their sales reports at the end of the week. The sales director subscribes to these reports, thereby receiving an update automatically (e.g., upon polling by the system). With each new instance or versions of previously subscribed objects, the system automatically updates the sales director's summary sheet. The director then sends the updated summary report, via publication, to the VP of Sales (and any other interested subscribers).

Detailed Description Text (73):
Shown in further detail in FIG. 3C, these tools will now be described. Mail button 361 allows a user to mail a data object, such as a page, a set of pages, an entire spreadsheet notebook, or other data object, to selected other users. Publish button 362 lets the user publish a data object to selected other users, set a version depth, issue new versions of published data objects, manage a subscriber list, and clear any current publication from the active notebook. Use button 363 lets the user insert mailed or published data objects into his or her own notebooks, use mailed notebooks, and delete objects from the Object Exchange module. Manage Pages button 364 lets the user see information on mailed and published pages inserted into the active notebook; rename, update, and change the update method of mailed and published pages individually; change the version of published pages individually, and remove mailed and published pages from the active notebook. Index button 365 displays a "workgroup index" (a special notebook page named "Workgroup") that contains information on the active notebook's current publication and all shared pages currently inserted into it. Address button 366 lets the user create and select address books, create and manage address groups, and manage addresses of workgroup desktop users contained in the address books and groups. The Poll button 367 instructs the Object Exchange module of the system to poll one or more (primary) messaging accounts.

Detailed Description Text (78):
Menu bar 381 includes an Accounts submenu (submenu 386 of FIG. 3E) for adding, deleting, and configuring messaging or transport accounts. The Account Status indicator 382a tells the user how many objects are waiting to be sent through each account. As shown, for instance, the Object Exchange dialog 380 of FIG. 3D has no (zero) objects waiting to be sent via MCI Mail. Polling Mode indicator 382b allows the user to specify a polling mode (e.g., automatic or manual) for each messaging account; if desired, polling may also be deactivated for particular accounts. The right side of the Object Exchange window 380--the Active Transport indicator 383--displays each step of the polling process. Screen indicators are provided for telling the user when the Object Exchange module is logging into an account,

querying an account for objects to transmit or pick up, sending objects, or logging out of an account. Indicator 383 includes a message box beneath the polling indicators for displaying screen messages of activity when the Object Exchange module polls an account. The Poll Now button 384, when activated by the user, instructs the Object Exchange module to poll the currently selected messaging account (as highlighted in 382a).

Detailed Description Text (79):
Object Exchange detail area 385 provides a list of all outgoing objects waiting to be sent by the Object Exchange. Alerts (system messages) are displayed in a drop-down list box. The bottom box shows the status of the Object Exchange outgoing queue and provides details about each object waiting to be sent out; details include the sender's user name and messaging service that the object is being sent through.

Detailed Description Text (96):
Once an Object Exchange account has been created, the system is ready to begin publishing and mailing data objects. Publishing and subscribing of data objects, such as spreadsheet notebook pages, establishes ongoing data-sharing relationships between one's own data objects and those of remote users. When a user publishes notebook pages and other workgroup members subscribe to them, the data in those pages appear in the subscriber notebooks exactly as it appears in the user's own notebook. After establishing a publication for a notebook, one can change the data in his or her pages or the set of pages he or she is publishing and issue new versions of the publication to subscribers. The subscribers in turn can decide to have those changes appear automatically in their notebooks, or to appear only when they request an update.

Detailed Description Text (109):
The user may "mail" a notebook, sending the entire notebook and all of its properties. Recipients receive all data values, cell formats, formulas, graphs, graphics, and the like. Mailing a notebook is useful when one wants to distribute a model or template, such as a form that should be filled out a certain way, to a number of users. The user can also mail selected pages from a notebook. This is done when one wants to send a set of data or a report to a group of users once, but does not need to update them later. As with publishing, mailing pages sends the data values and cell formats from the original notebook pages (but in a preferred embodiment does not send formulas, graphs, or graphics). Recipients can copy mailed data, referred to at end formulas, and redistribute it. Preferably, however, they cannot change it and save the changes, because mailed pages are read-only pages "owned" by the sender.

Detailed Description Text (111):
When a user publishes, issues versions, or mails data objects, the system submits the published or mailed objects to the Object Exchange, for holding them in its outgoing queues. When the Object Exchange polls an account, it transmits all the objects--notebooks, sets of pages, or individual pages--that the user has sent to that account, and picks up all the objects that the user has received to that account since the Object Exchange last polled it. The Object Exchange holds incoming objects in the user's Object Store, ready for use. Thus the Object Store holds copies of share data objects and, thus, behaves like a database of the shared objects sent to the user. The Object Store makes all of the user's shared objects continuously available for use in any notebook, whether or not the user is currently connected to messaging services. In a preferred embodiment, shared objects remain in the Object Exchange until deleted.

Detailed Description Text (114):
When a user inserts published pages, he or she becomes a subscriber. Subscribing establishes remote, dynamic links with the publisher's notebook. When the publisher issues new versions of the published pages, the user's Object Exchange

automatically receives them and responds accordingly to the update method specified by the user (i.e., automatic or manual updating).

Detailed Description Text (115):
Automatic updating automatically inserts new versions of published pages into the user's notebooks when his or her Object Exchange receives them. Manual updating, on the other hand, inserts new versions only when the user requests an update.

Detailed Description Text (117):
Unlike published pages, mailed pages are single-issue only. The user inserts mailed and published pages into his or her notebooks in the same way, but mailed pages cannot be updated. Mailed pages are useful when a workgroup member wants to send a set of data or a report once, but does not need to issue new versions of it later.

Detailed Description Text (118):
Referring now to FIG. 6A, a preferred interface and method for using published or mailed objects will now be illustrated. Initially, the user selects the Workgroup Use button 363 (from FIG. 3C). In response, the system displays Use Notebooks and Pages dialog box 610. The dialog 610 includes a Description list 611 describing all objects --pages, sets of pages, and notebooks--currently stored in the user's Object Exchange. As the user scrolls through the list, the bottom part of the dialog changes to show information on the selected object. The Contents field 612 shows the selected object's type (e.g., mailed pages, published pages, mailed notebook, or the like). Distributed By field 613 includes the address of the user who mailed or published the selected object. If the selected object is one of the user's own publication, "OBEX" (for Object Exchange) appears here. Last Distributed field 614 displays the data and time the last version of the selected object was published, or when it was mailed. For published pages having more than one version (i.e., the publisher has specified a version depth greater than 1), the user can select a particular version to insert through Version box 615. Update Method box 616, on the other hand, allows the user to select an update method (manual or automatic). Pages field 617 allows the user to select only some of the pages into the notebook. By default, all of the pages in a selected set are inserted into the notebook. The user selects or deletes objects using the buttons 618. Selected pages are inserted into the active notebook, using the first available empty, unnamed pages.

Detailed Description Text (119):
Referring now to FIG. 6B, a preferred interface and method for managing inserted pages will now be described. After the user has opened the notebook that contains the inserted pages, he or she selects the Manage Pages button 364 (from FIG. 3C). In response, the system displays Manage Inserted Pages dialog box 620. The dialog box 620 lists in Inserted Pages box 620, although the mailed and published page is inserted in the active notebook. As the user scrolls through the list, the right side of the dialog box changes to show information about each selected page. Description field 622 shows a description of the object the page came from. Distributed By field 623 shows the address of the person who mailed or published the page. Date Last Distributed field 624 shows the date the last version of the page was mailed or published. Status field 625 shows the update status of the selected page: "Dynamic" means the selected page uses automatic updating; "Current" means the selected page uses manual updating, with the latest version being inserted into the notebook; and "Out of date" means the selected page uses manual updating, and the latest version has not yet been inserted into the notebook. Original Name field 626 keeps track of the original name of the selected page. Version box 627 shows the version of the in selected page. By default, Current is the most recent version. To insert a previous version, the user selects Back. To enter the change, the user selects Update from buttons 629. Update Method box 628 shows whether the selected page is set to automatic or manual updating. This may be changed by the user as desired. To rename an inserted page, the user selects Rename from buttons 629. Likewise, to remove an inserted page, the user selects Remove

from buttons 629. The user concludes the dialog by selecting Close from buttons 630.

Detailed Description Text (122):
The Object Exchange lets users share information (objects) between different applications and different users of the same application. As previously described, a user can share notebooks, notebook pages, or sets of notebook pages from a spreadsheet. As another example, in a database environment users can share tables, query results, folders, and the like. The Object Exchange is preferably implemented not as an application in itself, but as a middle link between applications and the outside world. In other words, it interacts with the operating system and messaging services to provide workgroup and communication services to applications. The Object Exchange maintains a store of shared objects on a user's local computer, which makes shared data continuously available to the user without requiring that user to be continuously connected to a communications network. This object store lets a user use shared pages and notebooks even if a local area network is inaccessible.

Detailed Description Text (124):
The Object Exchange also supports version depth for publications. The version depth determines how many versions of a publication the Object Exchange holds for subscribers at any given time. Through dynamic links with objects stored in the Object Exchange, subscriber applications can support both manual and automatic updating. With automatic updating, changes to the original data that publishers make and issue appear automatically in the subscriber application. With manual updating, those changes appear in the subscriber application only when the subscriber requests them.

Detailed Description Text (185):
(3) LAN: Local area networks such as Novell NetWare, Banyan 5.0, Microsoft LAN Manager 2.1, AT&T StarGroup 3.5.1, IBM LAN Server 2.0, and 3Com 3+ and 3Com 3+ Open, or the like.

Detailed Description Text (234):
The ObxAccountSetTransportParam function updates the parameter szParameter in the account referenced by hAccount. szSetting specifies the new setting. The following table lists each valid setting for szParameter and szSetting:

Detailed Description Text (245):
The ObxUpdateAccount function updates the account referenced by hAccount. Changes made to an account using function calls (polling frequency, parameter settings, and so on) are not saved until this function is called. This function returns 1 if it succeeds or 0 if it fails.

Detailed Description Text (252):
The ObxABGetGroup function returns a handle to a group in the address book referenced by hAddressBook. nIndex is an integer indicating the placement of the group in the list of groups. For example, nIndex may be set to 1 to get the first group, 2 for the second, and so on. Group entries may be viewed, added, removed, or updated with the group handle. ObxGetAccountByName may be used to get a group handle using its name. This function returns NULL if it fails.

Detailed Description Text (253):
The ObxABGetGroupByName function returns a handle to a group in the address book referenced by hAddressBook. The string szGroupName specifies the name of the group to find. With the group handle group entries may be viewed, added, removed, or updated. This function returns NULL if it fails.

Detailed Description Text (314):
The ObxEntryUpdateAddress function updates the address referenced by hAddress in

the address book entry referenced by hEntry. Changes made to an address using function calls are not saved until this function is called. ObxGroupGetBlankEntry, ObxGroupGetEntry, or ObxGroupGetEntryByAlias sets hEntry and ObxEntryGetBlankAddress, ObxEntryGetAddress, or ObxEntryGetAddressByDescription to set hAddress before calling this function. This function returns 1 if it succeeds or 0 if it fails.

Detailed Description Text (317):
The ObxGroupGetEntry function returns a handle to an entry in the address book group referenced by hGroup. nIndex is an integer indicating the placement of the entry in the list of groups. For example, nIndex to 1 to gets the first entry, 2 for the second, and so on. With the entry handle one can view, add, remove, or update addresses in the entry. ObxGroupGetEntryByAlias is used to get an entry handle using its alias. This function returns NULL if it fails.

Detailed Description Text (318):
The ObxGroupGetEntryByAlias function returns a handle to an entry in the address book group referenced by hGroup. The string szAlias specifies the alias to be used by the entry. With the entry handle one can view, add, remove, or update addresses in the entry. This function returns NULL if it fails.

Detailed Description Text (319):
The ObxGroupUpdateEntry function updates the entry referenced by hEntry in the group referenced by hGroup. Changes made to an entry using function calls are not saved until this function is called. This function returns 1 if it succeeds or 0 if it fails.

Detailed Description Text (433):
ObxObjectUpdateSubscribers is used to update the list and to send new subscribers the current version of the publication. ObxObjectUpdateSubscribersNextVersion is used instead when sending the next version of the publication to new subscribers. Finally, ObxPubFormSetSubscribers saves the subscriber list.

Detailed Description Text (440):
When a subscription is cancelled or a subscription offer is rejected, the subscriber list is automatically updated in the publisher's object stores. When this occurs, ObxSubscribersNotifyOnUpdate specifies a procedure to call. The callback procedure specified can then get the subscriber list and check it for changes.

Detailed Description Text (442):
Subscriber lists store addresses and information on the status of each subscriber (whether or not an offer has been accepted or rejected, whether or not the subscription has been cancelled). To get the subscriber list, a handle to the object containing the subscriber list is opened. Next, ObxObjectGetSubscribers gets a handle to the subscriber list. The following functions may be used to review subscriber information: ObxSubscribersGetAddress, ObxSubscribersGetCountOf, ObxSubscribersGetOffer, ObxSubscriberGetOfferFromName, ObxSubscribersGetStatus, and ObxSubscribersGetStatusFromName.

Detailed Description Text (451):
Subscribers who decline a subscription offer or cancel their subscription are automatically removed from the subsciber list. ObxObjectGetSubscribers gets a handle to a sub to remove subscribers from a list manually. ObxSubscribersEmpty removes all addresses from the list. ObxSubscribersGetAddress removes individual addresses then ObxSubscribersRemove removes the address. This step is repeated for each address to remove. ObxObjectUpdateSubscribers updates the subscriber list, and ObxPubFormSetSubscribers saves the revised subscriber list.

Detailed Description Text (453):

Subscribing lets one user use another OBEX user's data without the worry of not receiving updates. Subscription offers are available, eliminating the need to subscribe to unwanted data.

Detailed Description Text (501):
Once a connection is established with OBEX the following functions are used to check its status: ObxAreasInfoGetCount gets the number of OBEX areas in the user's object store. ObxAreasInfoGetDescriptions gets the descriptions of all the OBEX areas in the user's object store. ObxAreasInfoGetNames gets the names of all the OBEX areas in the user's object store. ObxAreasInfoNotifyOnUpdate specifies a callback procedure to call when the specified AREASINFO changes. ObxGetAccountBeingPolled gets the name of the account being polled. ObxGetAreasInfo gets a handle used to reference information about all OBEX areas. ObxGetCountOfAccounts gets the number of accounts that exist in OBEX. ObxGetCountOfAreas gets the number of areas in OBEX. ObxGetErrorCode gets the error code of the last error that occurred in OBEX. ObxGetErrorText gets text describing the last error that occurred in OBEX. ObxGetTotalOutgoing gets the total number of outgoing objects (declarations, mailed objects, published objects). ObxIsPolling checks if OBEX is currently polling accounts. The following functions specify procedures to call when an aspect of OBEX changes: ObxAreaTocNotifyOnUpdate specifies a callback procedure to call when the specified AREATOC changes. ObxAreasInfoNotifyOnUpdate specifies a callback procedure to call when the specified AREASINFO changes. ObxObjectNotifyOnUpdate specifies a procedure to call when an object is updated. ObxSubscribersNotifyOnUpdate sets a procedure to call when a subscriber list is updated.

Detailed Description Text (507):
FIG. 8 summarizes a workgroup system 800 of the present invention. The system comprises a plurality of workgroup members 810, 820, 830, 840 connected together through any convenient messaging system. Each member has his/her own (private) Object Exchange with an Object Manager which effects actions by posting messages or "forms" to either the local Object Exchange (assuming one is the publisher) or the Object Exchange of others (subscribers). Connectivity for the system is provided by the various Object Exchange engines negotiating forms. Each Object Exchange posts and retrieves forms at polling intervals (which may be set to continuous polling). In this fashion, the system provides "deferred connectivity," allowing members to contribute to the workgroup--freely exchanging data objects--without being constrained in place or time. The protocol of distributing objects is, therefore, one of negotiating forms between various Object Exchange engines.

Detailed Description Text (519):
A version of the publication may now be issued. If a subscriber has linked his or her application (client process) to the Object Exchange through a DDE "hot link" (see Petzold supra for detailed description), when a new version is picked up the subscriber will automatically be updated. As shown in FIG. 9D, the Object Exchange receives issue command 940 to issue a new version as follows:

Detailed Description Text (532):
While the invention is described in some detail with specific reference to a single preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. For instance, although the foregoing example has focused on sharing spreadsheet pages, the type of data sent is not restricted to any particular data type. Instead, the present invention may be applied to any desired granularity of data object, including a spreadsheet notebook, a database table, a bitmap graphic, or other data "object." Thus, the true scope of the present invention is not limited to any one of the foregoing exemplary embodiments but is instead defined by the appended claims.

Detailed Description Paragraph Table (28):

String Messaging service
MCI MCI Mail MHS Novell's NetWare Message Handling Service (MHS), version 1.5, or its upgrade, NetWare Global Messaging (NGM) LAN Local area networks such as Novell NetWare, Banyan 5.0, Microsoft LAN Manager 2.1, AT&T StarGroup 3.5.1, IBM LAN Server 2.0, and 3Com 3+ and 3Com 3+ Open. MAPI Microsoft's Mail Application Programming Interface. ccMAIL Lotus cc:Mail and other VIM-compliant messaging services. VIM is Vendor Independent Messaging. WPO WordPerfect Office, version 4.0a or later. NOTES Lotus Notes, version 3.0 or later.

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: (synchroniz$ or updat$) near (voice near messag$)

Display: 50 **Documents in Display Format:** - **Starting with Number** 1

Generate: O **Hit List** ◉ **Hit Count** O **Side by Side** O **Image**

Search Clear Interrupt

Search History

DATE: Saturday, February 14, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L27	L25 and (message near box)	1	L27
L26	L25 and (subscriber near information)	3	L26
L25	(synchroniz$ or updat$) near (voice near messag$)	26	L25
L24	L22 and (message near box)	2	L24
L23	L22 and (subscriber near directory)	0	L23
L22	L20 and (schedul$)	53	L22
L21	L20 and (mail near schedul$)	0	L21
L20	L19 and synchroniz$	66	L20
L19	L17 and (subscriber near information)	92	L19
L18	L17 and (updat$ near shar$)	21	L18
L17	l1 and (updat$ near request$)	551	L17
L16	L14 and (message near box)	21	L16
L15	L14 and (message near box)	21	L15
L14	L13 and server	407	L14
L13	L12 and (updat$ or synchroniz$)	574	L13
L12	L11 and database	689	L12